

Paper 040-2011

JMP® 9 Add-Ins: Taking Visualization of SAS® Data to New Heights

Eric Hill, SAS Institute Inc., Cary, NC

ABSTRACT

For several years, JMP® has been a great tool to use for discovering relationships in your SAS® data. JMP 9, released last fall, vastly extends the analytic and discovery capabilities that were previously available, including:

- many improvements to the interactive graphical discovery tool, Graph Builder
- the ability to automatically place maps behind geographical data
- the ability to reach out to R to execute analytical methods

Combine those improvements with the ability to create add-ins for JMP that can be deployed easily throughout the enterprise and you have a tool that can take statistical discovery and information visualization to new heights in your organization.

INTRODUCTION

JMP is an interactive statistical visualization and discovery tool that was developed at SAS and has been shipping since 1989. With the release of JMP 7.0 in 2007, the ability of JMP to access the SAS data in your organization was improved dramatically. As a result, it is much easier for JMP users to discover relationships in their SAS data. JMP 9.0 (released in October, 2010) has added a number of features that will facilitate using JMP to make discoveries and to distribute the fruits of those discoveries to other JMP users.

We will start with some SAS data from a fictional retail movie distributor, FlickBuster. The FlickBuster data consists of three SAS data sets: Customers, Movies, and Rentals. The Rentals data set contains transaction data on roughly 500,000 movie rentals by its customers between 2006 and 2010. We will use JMP to explore this data interactively, and then we will package up what we have discovered into a JMP Add-In that we can easily distribute to our colleagues. Using this add-in, we can show our colleagues what we found and encourage these users to explore further on their own.

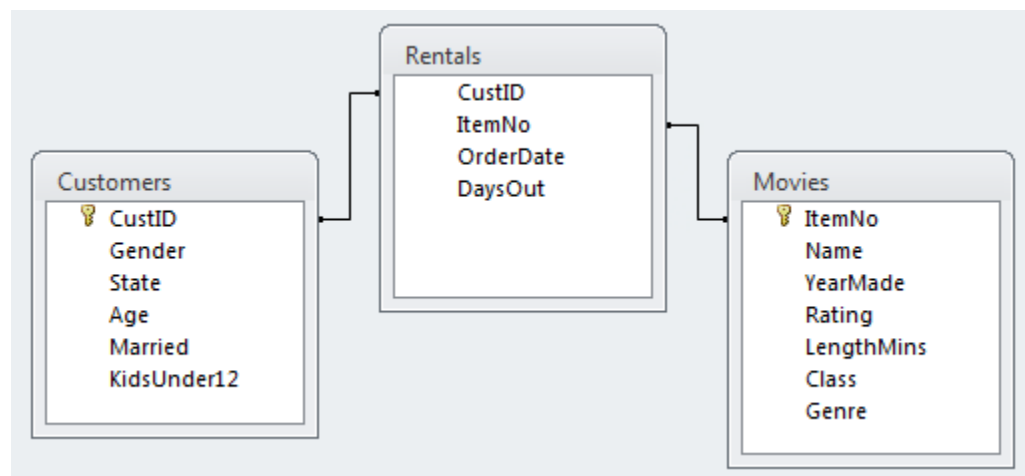


Figure 1. FlickBuster Data Relationships

MAPPING FLICKBUSTER RENTALS

The first step in any analysis, whether programmatic or interactive, is to prepare the data. In this case, we want to look at our movie rentals by state. For that analysis, we will need to create a view of our rental data that includes the details about the customer and the movie that we might be interested in. This view is something that a database administrator may have done on our behalf in many organizations, but just in case we're not that lucky, we will do the query ourselves. Using the JMP Scripting Language (JSL), we can connect to a SAS server, submit the necessary PROC SQL code to create the summary table, and import it into JMP for analysis.

```
fb_sas = SASConnect ( "SASApp" );
```

<Paper title>, continued

```
fb_sas << submit("\[
%include "\\jmpi\source\users\Hill\SGF2011\FlickBuster\SASCode\libdef.sas";

proc sql;
  create view flkbstr.rental_details_view as
    select r.CustID, r.ItemNo, r.OrderDate, year(r.OrderDate) as
OrderYear,
          r.DaysOut, m.Rating, m.LengthMins, m.Class, m.Genre,
          c.Gender, c.State, c.Age, c.Married, c.KidsUnder12
    from flkbstr.rentals r, flkbstr.movies m, flkbstr.customers c
    where r.CustID = c.CustID and r.ItemNo = m.ItemNo;

  create table work.state_summary as
    select distinct State, OrderYear, Class, Genre, Rating, Married,
KidsUnder12, count(*) as count
    from flkbstr.rental_details_view
    group by State, OrderYear, Class, Genre, Rating, Married,
KidsUnder12;
quit;

]\");

dtStateSummary = fb_sas << import data( "work.state_summary" );
```

You can use the Query Builder feature in SAS® Enterprise Guide® to build PROC SQL queries (like the one in the previous code) interactively. After the code is built in SAS Enterprise Guide, you can copy the PROC SQL code and paste it into JMP. The previous code takes about 20 seconds to run on the Rentals data set that contains 500,000 records. After this code runs, we have the following data ready for analysis:

State	OrderYear	Class	Genre	Rating	Married	KidsUnder12	count	
1	AL	2006	Classics	Action	NR	0	0	2
2	AL	2006	Classics	Action	NR	1	0	1
3	AL	2006	Classics	Action	NR	1	1	8
4	AL	2006	Classics	Action	PG	0	0	2
5	AL	2006	Classics	Action	PG	1	0	7
6	AL	2006	Classics	Action	PG	1	1	4
7	AL	2006	Classics	Comedy	PG	0	0	5
8	AL	2006	Classics	Comedy	PG	1	0	11
9	AL	2006	Classics	Comedy	PG	1	1	10
10	AL	2006	Classics	Comedy	R	0	0	3
11	AL	2006	Classics	Comedy	R	1	0	1
12	AL	2006	Classics	Comedy	R	1	1	4
13	AL	2006	Classics	Drama	NR	0	0	3
14	AL	2006	Classics	Drama	NR	0	1	1
15	AL	2006	Classics	Drama	NR	1	0	7
16	AL	2006	Classics	Drama	NR	1	1	4
17	AL	2006	Classics	Drama	PG	0	0	2
18	AL	2006	Classics	Drama	PG	1	0	8
19	AL	2006	Classics	Drama	PG	1	1	4

Figure 2. Summarized Rental Data Ready to Be Analyzed

<Paper title>, continued

To begin the analysis of movie rentals by state, we will use the **Graph Builder** platform in JMP. This tool is the first item in the Graph menu. Graph Builder was new in JMP 8.0. It is a drag-and-drop interface for experimentally creating graphical views of data. You simply drag columns from your data onto the various drop zones in Graph Builder, and Graph Builder will show you what it believes is the graph you want to see. When you launch Graph Builder, it appears similar to Figure 3.

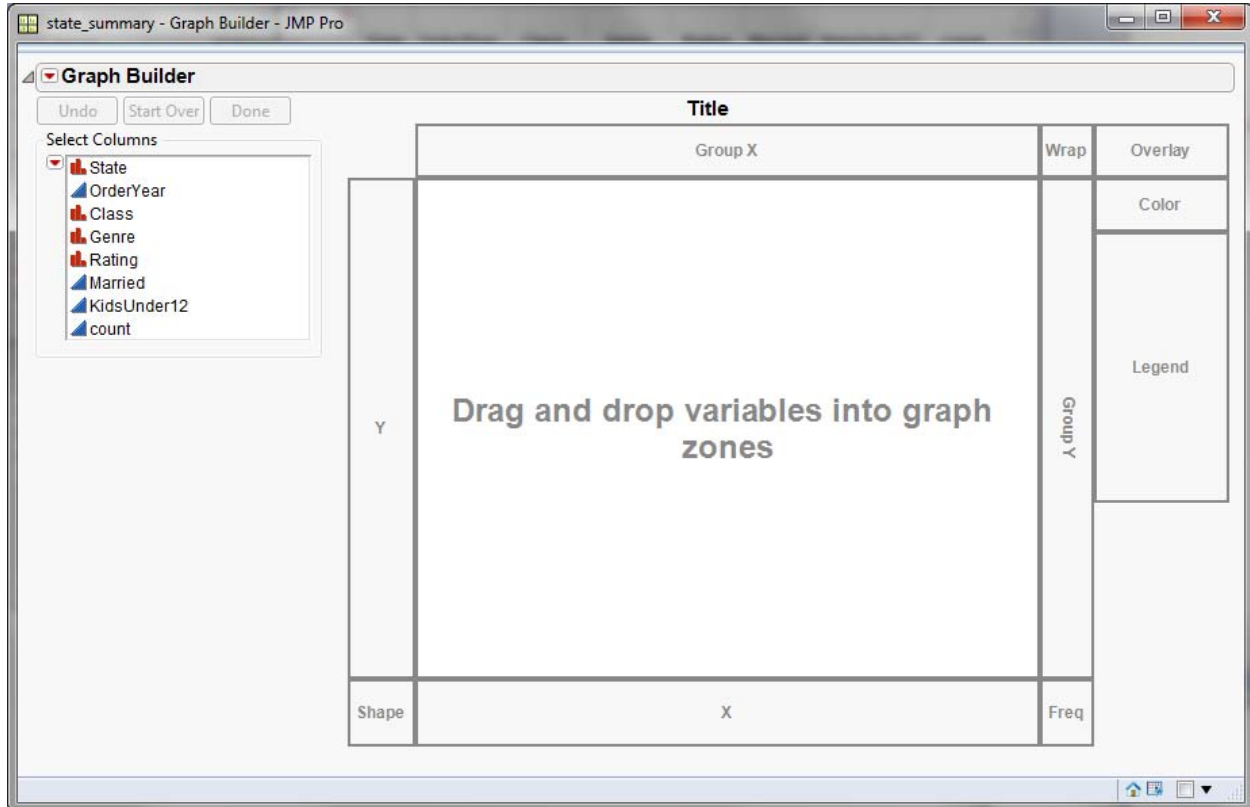


Figure 3. Graph Builder Platform in JMP

<Paper title>, continued

In JMP 9.0, Graph Builder got smarter about noticing that certain columns look like geographical locations, such as states. If we simply drag the State column from the column list over to the **Shape** drop-zone at the lower-left of the chart and then drag the count column and drop it into the large drop-zone in the middle, the result is displayed in Figure 4. JMP chose to display a heat map in the shape of the United States.

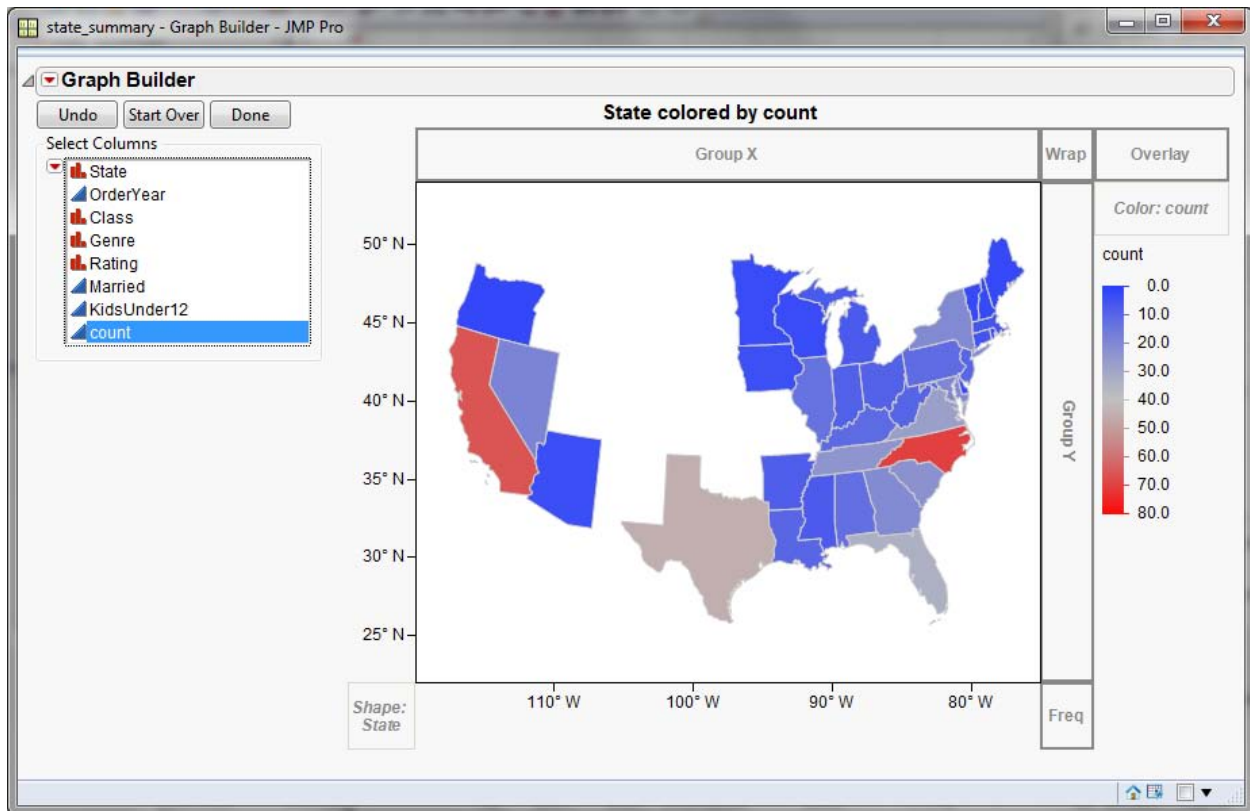


Figure 4. Heat Map by State of Movie Rental Volume

Notice that some of the states are not displayed at all. By default, states with no rental activity do not appear. Also, by default, Graph Builder determines the color of a state by *averaging* (rather than summing) the values in the count column. Lastly, if the blue-to-white-to-red gradient of the default heat map is not to your liking, you can choose a different built-in gradient or create a custom gradient by right-clicking the scale at the right. Adjusting all of those things and clicking **Done** results in Figure 5.

<Paper title>, continued

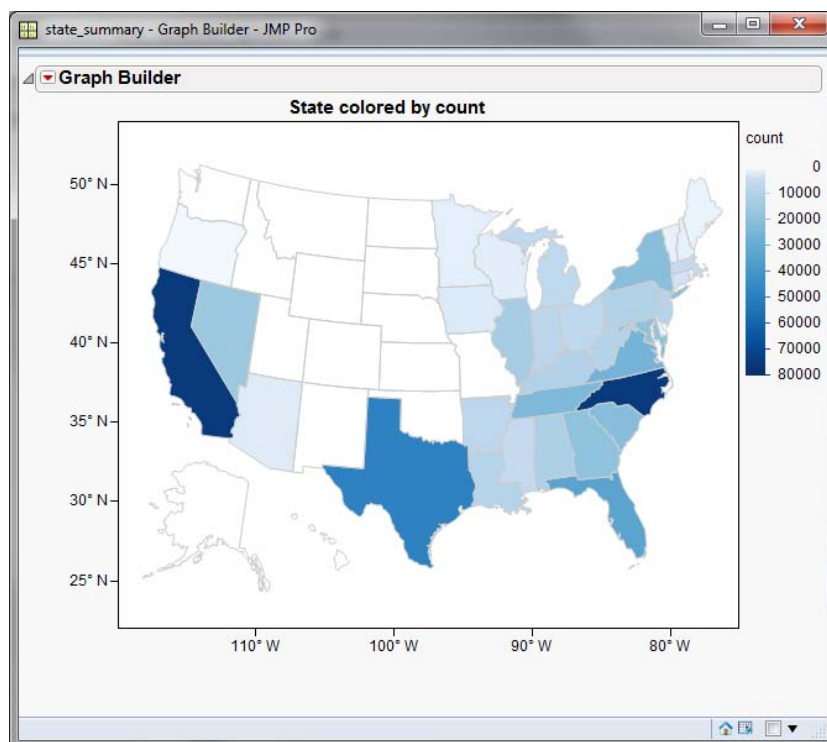


Figure 5. Customized Heat Map by State

Now that you've gone through the manual steps to get the visualization exactly the way you want it, you may want to grab the JMP script that would recreate it so that you can either reproduce it again later or provide it to others to assist them in creating this visualization. JMP platforms make this easy. If you click the red-triangle at the upper-left, and from the menu, select **Script > Save Script To Script Window**, you will have the JSL script that you can use to recreate that graph. We will use a form of that script later in the JMP Add-In when we use the add-in to distribute information to our JMP colleagues.

ANALYZING RENTAL RETURN DELAY

Another statistic that it would be valuable for FlickBuster to analyze is the amount of time it takes for customers to return the movies that they rent. To begin the investigation, we run the following JSL script in JMP. This script submits some PROC EXPORT code to SAS to copy the entire Rental_Details data set over to JMP. Then the script applies some other JMP settings to the columns to get the data ready for analysis in JMP.

```
fb_sas << submit( "[
  proc export data=flkbstr.rental_details
    outfile="\\server\FlickBuster\Jmp_gen\rental_details.jmp"
    dbms=jmp
    replace;

  run;
]" );

dtDetails = Open( "\\server\FlickBuster\Jmp_gen\rental_details.jmp" );
dtDetails:CustID << Set Modeling Type( Ordinal );
dtDetails:ItemNo << Set Modeling Type( Ordinal );
dtDetails:OrderYear << Set Modeling Type( Ordinal );
dtDetails:Married << Set Modeling Type( Ordinal );
dtDetails:Married << Value Labels( {0 = "No", 1 = "Yes"} );
dtDetails:KidsUnder12 << Set Modeling Type( Ordinal );
dtDetails:KidsUnder12 << Value Labels( {0 = "No", 1 = "Yes"} );
```

If your SAS server can write to a network share that your JMP session can access, PROC EXPORT is the fastest way to get data from SAS to JMP.

<Paper title>, continued

Once again, we will begin the analysis interactively. In this example, we will use the Distribution platform in JMP. The column that records the duration of each rental is DaysOut. Using the Distribution platform, you can easily get linked histograms for DaysOut, LengthMins, Married, and KidsUnder12, which are factors that you might suspect affect rental duration. Selecting the **Yes** bar in the KidsUnder12 histogram suggests a relationship between having young children and the amount of time it takes to return movies (see Figure 6).

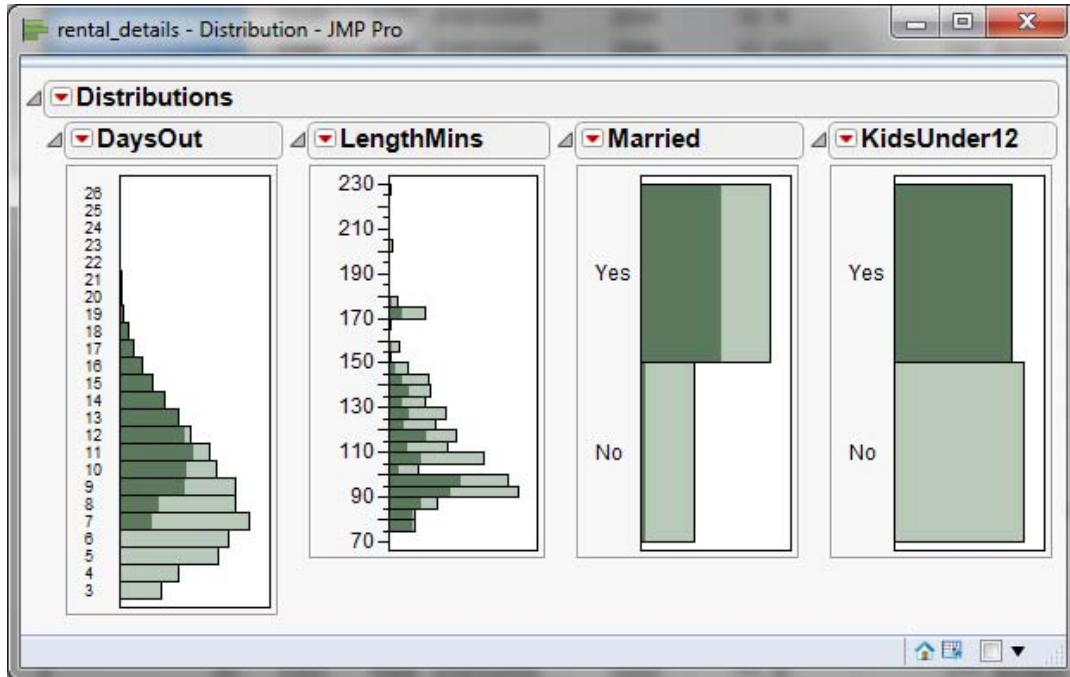


Figure 6. Distribution of Factors Affecting Rental Duration

Based on this discovery, you can then fit a regression model for DaysOut using the **Fit Model** platform in JMP. . Select **Analyze > Fit Model** to open the Fit Model dialog box. Use this dialog box to construct the model to fit (see Figure 7).

<Paper title>, continued

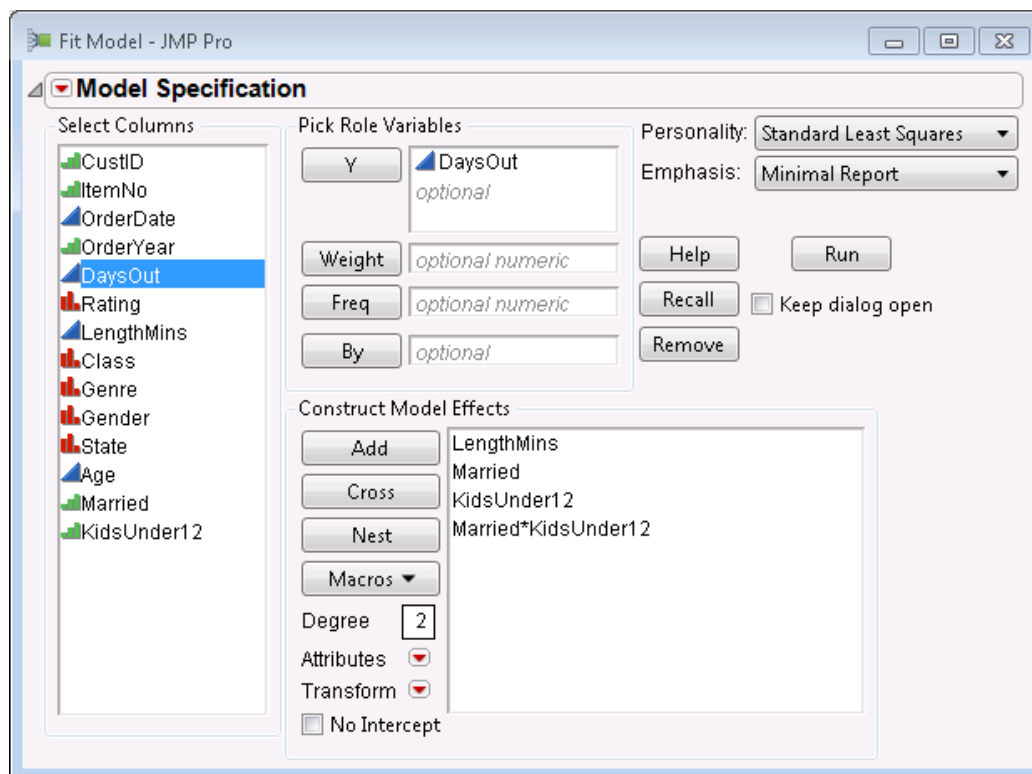


Figure 7. Fit Model Dialog Box

Clicking **Run** computes the regression and produces parameter estimates (see Output 1).

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob> t
Intercept	-1.87534	0.013189	-142.2	<.0001*
LengthMins	0.0714849	0.000105	683.85	<.0001*
Married[Yes-No]	0.0087311	0.006857	1.27	0.2029
KidsUnder12[Yes-No]	7.4964077	0.014912	502.73	<.0001*
Married[Yes-No]*KidsUnder12[Yes-No]	-1.915694	0.016062	-119.3	<.0001*

Output 1 Regression Parameter Estimates

However, looking at a table of parameter estimates is not the best way to visualize the effects of the independent variables on the dependent variable in JMP. A much better way is to use the **Factor Profiler** in JMP. The profiler can be opened by clicking the red triangle at the top of the regression report. Figure 8 shows two snapshots of the factor profiler for the DaysOut regression model.

<Paper title>, continued

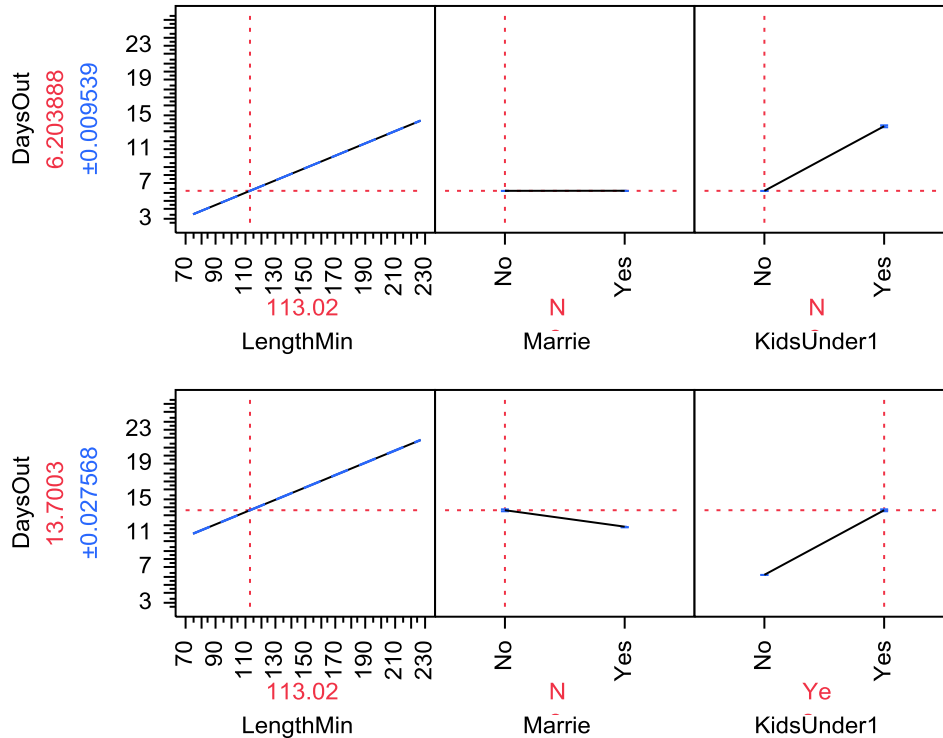


Figure 8. Snapshots of the Factor Profiler for the DaysOut Regression Model

The difference between the two snapshots is that the crosshairs in the KidsUnder12 plot have been moved from **No** to **Yes** in the lower snapshot. Both snapshots clearly show the positive relationship between the length of a movie and the number of days the customer keeps it. The snapshots also show that customers with young children tend to take longer to return movies than those without children. The change in the slope of the line in the Married plot shows the interaction between Married and KidsUnder12. The profiler appears to show that single customers with young children keep their movies longer than married customers with young children.

Something else to consider doing after you have found a regression model you like is to save the prediction formula from the regression model as a formula column in the data table. You can access this option by once again clicking the red triangle at the top of the regression report. From the menu, select **Save Columns > Prediction Formula**. After you have saved the prediction formula in the data table, you can run the standalone profiler on that data table without having to go back through Fit Model. The standalone profiler can be invoked by selecting **Graph > Profiler**.

The length of time customers keep movies before returning them is an important statistic for FlickBuster, so this is a visualization we will want to make available in our JMP Add-In.

USING R TO SHOW GENRE TRENDS

Another question that the management of FlickBuster might ask is whether there are any notable trends in the genres of movies that customers are renting. While there are many ways in JMP that you might go about analyzing such a trend, we will highlight the integration of the R Programming Language, which is a new feature in JMP 9. The sparkTable package (one of the available R packages) creates HTML tables containing sparklines (see Figure 9).

<Paper title>, continued

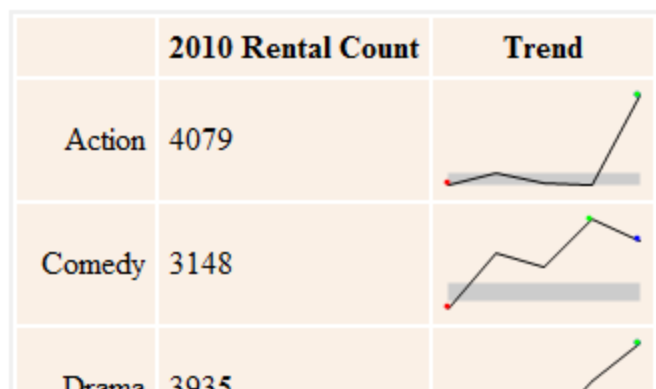


Figure 9. Example HTML Table with Sparklines

In order to use this sparkTable package, we first need to generate summary data in the format that the package requires. For that, we will turn once again to PROC SQL:

```
proc sql;
  create table work.genre_counts as
  select distinct OrderYear, genre, count(*) as count
  from flkbstr.rental_details
  ^WHERE ^AGE ^AGEAND ^MARRIED ^MARRIEDAND ^CHILDREN
  group by genre, OrderYear
  order by genre, OrderYear;
quit;
```

Notice that the WHERE clause in the PROC SQL code does not look correct. For the subset of rentals that we want to study, we are going to use JMP to prompt the user for the age range, marital status, and children status. We will use the following JSL code to produce the dialog box in Figure 10. Then based on user responses, we will replace the placeholders in the PROC SQL code and submit the code to SAS to produce the data in the form we need.

```
::customNewWin = New Window( "SparkLines Report",
  H List Box(
    V List Box(
      Panel Box( "Select an age demographic:",
        rbAge = Radio Box( {"18 to 34", "35 to 54",
          "55 and up", "All"} )
      ),
      Panel Box( "Select marital status:",
        rbMaritalStatus = Radio Box( {"Single", "Married",
          "Either"} )
      ),
      Panel Box( "Has Children Under 12?",
        rbChildren = Radio Box( {"Yes", "No", "Either"} )
      )
    ),
    Panel Box( "Generate SparkLine Report",
      Lineup Box(
        N Col( 1 ),
        Button Box( "Run", ::onRun ),
        Button Box( "Cancel",
          ::customNewWin << CloseWindow;
          Throw();
        ),
        Text Box( " " ),
        Button Box( "Help", ::HelpScript )
      )
    )
  );
```

<Paper title>, continued

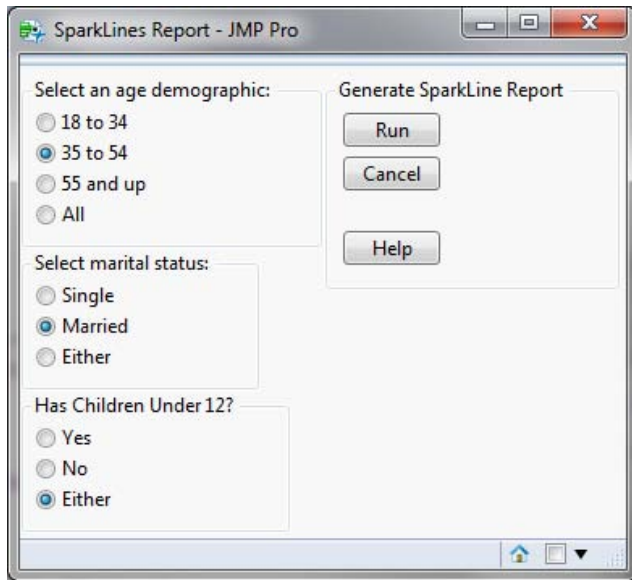


Figure 10. Resulting Custom Dialog Box

After prompting the user and asking SAS to generate the summary data we need, we can get the data back to JMP either using PROC EXPORT (as we did earlier), or if we do not have the option of a network share for SAS to place the exported JMP data on, the Import Data JSL function available in JMP can import the SAS data. In this case, the SAS data that is coming over is very small. We will use the **Invisible** option of the **Import Data** function so that the user doesn't see the data table itself, just the report that it produces.

```
dtGenre = fb_sas << Import Data( "work.genre_counts", Invisible );
```

Now that the data is in JMP, we can look at initializing the R environment from JMP, pushing our data over to R, submitting the R code that we need to generate the sparkline HTML table, and then retrieve the results in JMP. This process is achieved with the following JSL script.

<Paper title>, continued

```

// Initialize the R environment
R Init();
if ( R Is Connected(),
    // Check for the existence of the package we need
    libCheck = R Submit( "library(sparkTable)" );
    if ( libCheck == -1,
        throw( "The sparkTable R Package is not available." );
    );
);

// Send a JMP data table to R as an R data frame
R Send( dtGenre, RName("genre_counts" ) );

// Submit R code to use the sparkTable package
success = R Submit( "\[
    meta <- sparkTable_Config(
        genre_counts,
        groups = c( "Action", "Comedy", "Drama", "Family", "Horror",
                    "Musical", "RomCom", "Romance", "Thriller", "War", "Western" ),
        groupVar="Genre",
        vars=c("count"),
        typeNumeric=c("last"),
        typePlot="line",
        output="html")
    # this meta object can now be used to generate an EPS-output in R
    rowVec <- c( "Action", "Comedy", "Drama", "Family", "Horror",
                "Musical", "RomCom", "Romance", "Thriller", "War", "Western" )
    colVec <- c("2010 Rental Count", "Trend")
    html.text = print.sparkTable( meta,
        outdir="\\\\server\\SGF2011\\FlickBuster\\HTML",
        outfile="junk.html",
        rowVec = rowVec,
        colVec = colVec )
]\ " );

// If the R submit worked, get the results back into JMP
if ( success == 0, htmlText = R Get( "html.text" ) );

```

With a few additional lines of code (not shown) to embed the generated HTML table into some boilerplate HTML source that includes a CSS table style, the result (which is truncated for space) appears in Figure 11.

<Paper title>, continued

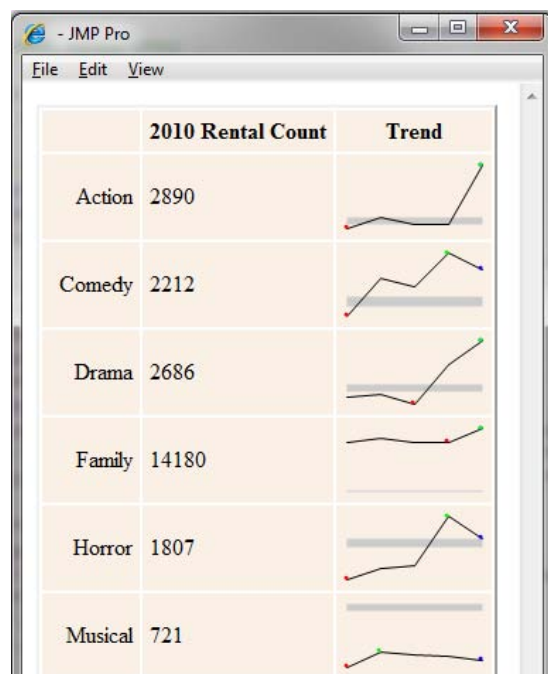


Figure 11. HTML SparkLine Table Generated by R

The end result is a static graphic (which is unlike what you expect when working with JMP), but it is nevertheless a graphic that would have taken more work to generate through JMP scripting. This is an example of using JMP as your *analytical hub*: using JMP to get data from SAS, performing analysis in R, and displaying the results in JMP.

BRINGING IT ALL TOGETHER IN A JMP ADD-IN

To this point, we have used JMP interactively to create interesting analyses from which we've captured the scripts necessary to reproduce the analyses, and we've also developed an analysis (the SparkLines table) completely via JSL scripting. With JMP 9, you can distribute the resulting scripts to others in your organization by creating a JMP Add-In. A JMP Add-In consists of the following parts:

- Some JSL scripts that perform the actions that you want the add-in to perform
- Other necessary JMP or SAS files, such as data tables or journals, SAS programs, or SAS data sets (optional)
- A text file named **addin.def** that defines at minimum a unique ID for the add-in
- An **addin.jmpcust** file containing the menu and toolbar customizations for the add-in

The **addin.def** file is a simple text file containing just a few name-value pairs that uniquely identify the add-in to JMP and optionally provides a user-friendly name that users may see. Here are the entire contents of the addin.def file for the add-in that we are developing:

```
id="com.jmp.eric.FlickBuster"
name="SAS Global Forum 2011 FlickBuster Add-In"
```

<Paper title>, continued

The value for “id” is a unique identifier for this add-in. This identifier is used internally by JMP. Because of the uniqueness requirement, the convention of using reverse domain name system (or Reverse-DNS) for naming add-ins is recommended, though not required. (Reverse-DNS involves starting from an Internet domain name (such as jmp.com) that is known to be unique to your company, reversing the parts, and adding to it). The value for “name” is the display name that the user will see for this add-in. There are other name-value pairs that can appear in the addin.def file, including the location of the add-in if it is not the same as the location of the addin.def file, but these name-value pairs are less frequently used than “id” and “name.”

Here are the contents of the directory containing the JSL scripts and the addin.def file for the add-in that we are creating:






Name	Size	Date modified
 a1_state_rental_heat_map.jsl	2 KB	1/27/2011 9:53 AM
 a2_DaysOut_Analysis.jsl	1 KB	2/1/2011 10:50 AM
 a3_genre_spark.jsl	6 KB	2/1/2011 10:16 AM
 addin.def	1 KB	2/1/2011 11:03 AM
 sas_connect.jsl	1 KB	1/27/2011 8:56 AM

Figure 12. Directory Containing Add-In Files

Notice that there is no **addin.jmpcust** file yet. We will use the new **Menu Editor** in JMP 9.0 to generate that file.

At this point, we can register this add-in in JMP. To do that, we will select **File > Open** from JMP, navigate to the directory containing the add-in files, select the **addin.def** file, and click **Open**. JMP will ask if we want to install the add-in named “SAS Global Forum 2011 FlickBuster Add-In,” at which point, we will click **Install**.

Since we have not yet created any menu or toolbar items to invoke our add-in’s scripts, having the add-in registered is not terribly exciting, but it does help us use the Menu Editor to generate the custom menu items that we want. To start the Menu Editor, select **View > Customize > Menus and Toolbars**. Initially, the Menu Editor comes up pointing at the current user’s menu customizations. But we want to create customizations that are specific to our add-in. To do that, click **Change** at the top of the Menu Editor window to open the **Change Customization Set** dialog box. In the **Customization Set To Modify** section of the dialog box, you can select the **JMP Add-In** option and then find your registered add-in from the drop-down list. (See Figure 13.)

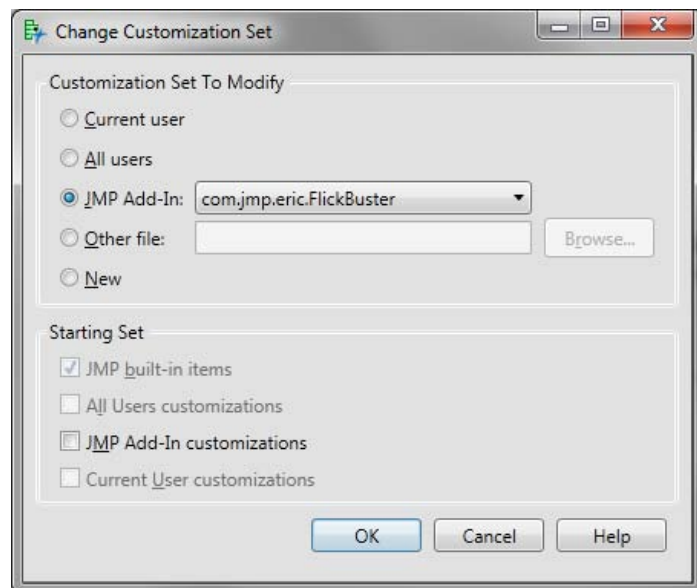


Figure 13. Change Customization Set Dialog Box for the Menu Editor

<Paper title>, continued

In most cases, the **Starting Set** for a JMP Add-In's customizations should include only **JMP built-in items** as pictured in Figure 13.

Figure 14 shows what the Menu Editor looks like after we have added our items. For each menu item, we checked the **Run JSL in this file** option, made sure the correct add-in was selected from the **Use add-in home folder** drop-down list, and used the **Browse** button to select the JSL file from the add-in's home folder to run for each item. The **\$ADDIN_HOME(com.jmp.eric.FlickBuster)** section of the file path is a JMP path variable that helps find files wherever an add-in happens to be installed, which can vary on different computers. Because we used the **Browse** button to select files, we did not have to type that path in.

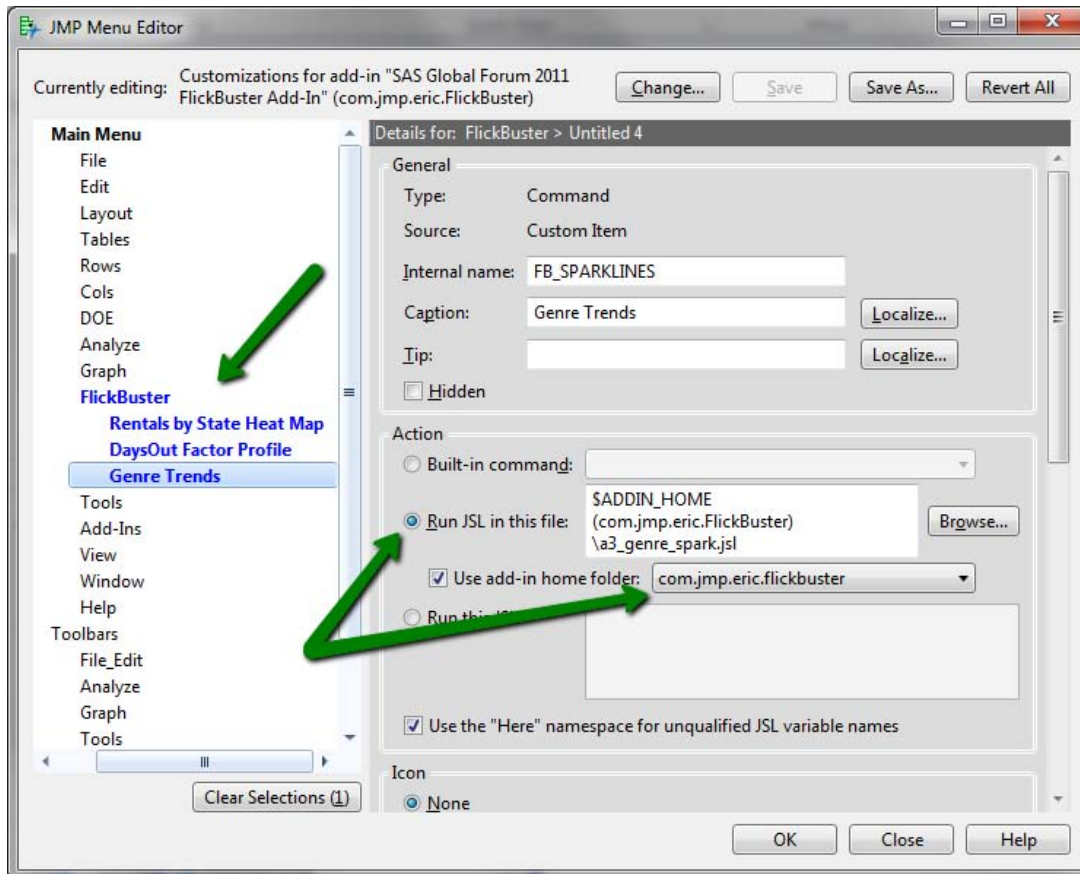


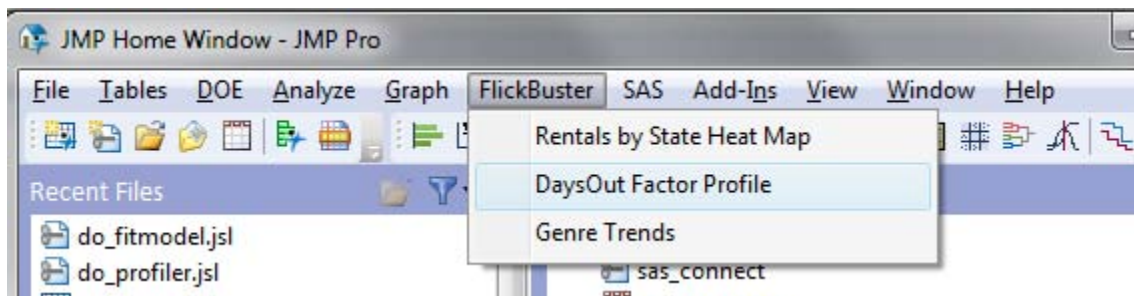
Figure 14. Adding Menu Customizations for the Add-In

After saving and closing the JMP Menu Editor, you will see an additional file in the directory that contains your add-in files:

 addin.jmpcust 2 KB 2/1/2011 12:39 PM

Figure 15. The addin.jmpcust file

Also, the customizations that we made for the add-in are now shown in the JMP main menu:



<Paper title>, continued

Figure 16. JMP Menu Showing Add-In Customizations

PACKAGING UP THE ADD-IN

Now that we have created menu customizations for our add-in, we are ready to package the add-in for distribution. In order to do that, you will need a tool capable of creating a ZIP file (such as WinZip). With WinZip, you can select the files that make up your JMP Add-In, right-click on these files in Windows Explorer, and select **WinZip > Add to Zip File**.

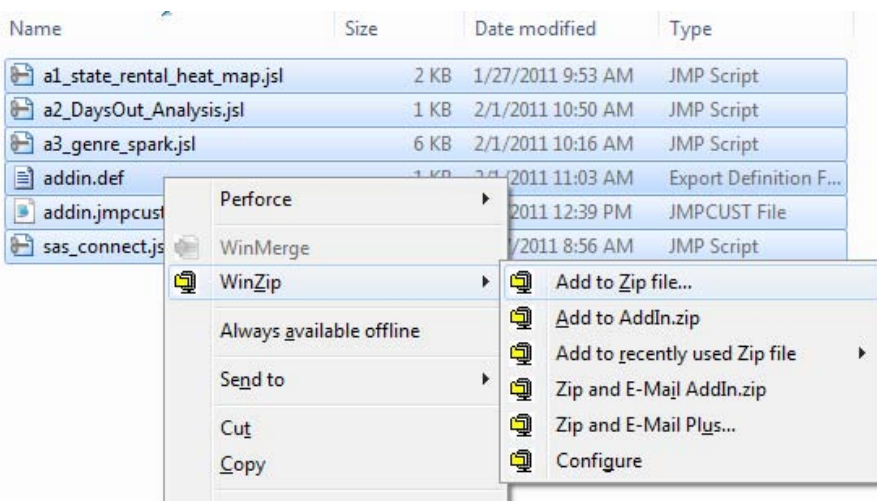


Figure 17. Creating the ZIP File for the Add-In

When creating the ZIP file, give the file the extension **.jmpaddin** instead of **.zip**. The **.jmpaddin** extension tells JMP that the ZIP archive contains a JMP Add-In, so that when the file is opened in JMP, JMP will attempt to install the add-in it contains.

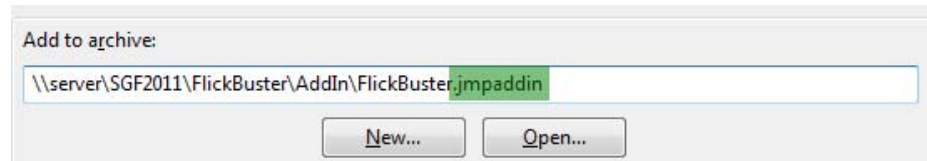


Figure 18. Using the .jmpaddin extension on the ZIP archive

Before testing the add-in, you should unregister the version that you registered earlier in order to create the custom menu item. Select **View > Add-Ins**, select your add-in, and click **Unregister**. Then, either use **File > Open** to open the **.jmpaddin** file you created, or simply double-click on the **.jmpaddin** file in Windows Explorer. JMP should prompt you to install the add-in, after which your custom menu items will appear in the menu.

After you confirm that your add-in works as intended, you can begin distributing the add-in (either via your corporate network or as an e-mail attachment) to other JMP 9.0 (or later) users..

CONCLUSION

JMP 9.0 continues the tradition of being at the leading edge of interactive statistical visualization and discovery, including visualizations using maps and the ability to connect to the R Programming Language. Combine those aspects with the ability to gain access to your SAS data and with the ability to create custom analysis packages as JMP Add-Ins for distribution across your organization, and you will definitely be taking the visualization of your SAS data to new heights.

<Paper title>, continued

ACKNOWLEDGMENTS

I would like to thank Xan Gregg and John Ponte for their work in JMP 9 bringing mapping capabilities to JMP in general and Graph Builder in particular. I would also like to thank Jeff Polzin and Kelci Miclaus, along with the SAS development team who developed the R extension for SAS that JMP is taking advantage of, for their work in bringing the ability to connect to the R Programming Language to JMP 9.

The sparkTable R package used in this paper was written and maintained by Alexander Kowarik, Bernhard Meindl, and Stefan Zechner. For more information, see the documentation for the sparkTable package at <http://streaming.stat.iastate.edu/CRAN/web/packages/sparkTable/index.html>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Eric W. Hill
SAS Campus Drive
SAS Institute Inc.

E-Mail: EricHill@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.